

AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

LISTING OF CLAIMS:

1. (Canceled)

2. (Currently Amended) A method according to Claim 1, ~~in which 21,~~
wherein the predefined number $\{L\}$ is variable from one predefined block of
instructions $\{\Pi_1, \dots, \Pi_N\}$ to another.

3. (Currently Amended) A method according to ~~one of Claims 1 to 2, in~~
~~which claim 21, wherein~~ the common block $(\Gamma(k,s))$ set of instructions comprises at
least one calculation instruction (γ_k) that is equivalent, vis à vis a covert channel
attack, to a calculation instruction of each predefined block (Π_1, \dots, Π_N) in the context
of a covert channel attack.

4. (Currently Amended) A method according to Claim 3, in which the
common block $(\Gamma(k,s))$ set of instructions also comprises an instruction to update a
loop pointer (k) indicating a number of executions already executed or performed
with the common elementary block $(\Gamma(k,s))$ set of instructions.

5. (Currently Amended) A method according to Claim 3 or ~~Claim 4~~, in which wherein the common block ($\Gamma(k,s)$) set of instructions also comprises an instruction to update a state pointer (s) indicating whether the predefined number (L_0) has been reached.

6. (Currently Amended) A method according to Claim 4 or ~~Claim 5~~, in which wherein the value of the loop pointer (k) and/or the value of the state pointer (s) are is a function of the value of the input variable (D_i) and/or of the number of instructions of in the selected block of instructions (P_j) associated with the input data value.

7. (Currently Amended) A method according to ~~one of Claims 1 to 6~~, in which claim 21, wherein, in order to successively effect several blocks of instructions chosen from amongst the ~~N~~ plural predefined blocks of instructions (Π_1, \dots, Π_N), each chosen selected block of instructions (Π_j) being is selected as a function of an input variable (D_i) associated with an input index (i), and the common elementary block ($\Gamma(k,s)$) set of instructions is executed a total number (L_T) of times, the total number (L_T) being equal to a sum of the predefined numbers (L_j) associated with each chosen selected block of instructions (Π_j).

8. (Currently Amended) A method according to Claim 7, during which wherein one and the same block of instructions may be chosen is selected several times according to the input variable associated with the input index (i).

9. (Currently Amended) A method according to ~~one of Claims 7 or 8, in which claim 7, wherein at least two of the following data items, (a) the value of the a loop pointer (k) and/or, (b) the value of the a state pointer (s) and/or, (c) the value of the input variable (D_i) and/or, and (d) the number of instructions of the selected block of instructions, (Pi) associated with the value of the input data item (D_i) are linked by one or more mathematical functions.~~

10. (Currently Amended) A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i, ~~the method comprising the following steps:~~

Initialisation:

$$R_0 \leftarrow 1; R_1 \leftarrow A; i \leftarrow M-1$$

As long as $i \geq 0$, repeat the common block ~~(F(k, s)) set of instructions:~~

$$k \leftarrow (I/s)x(k+1) + sx2x(/D_i)$$

$$s \leftarrow (k \bmod 2) + (k \text{ div } 2)$$

$$\gamma(k,s): \quad R_0 \leftarrow R_0 \times R_k \bmod 2$$

$$i \leftarrow i - s$$

Return R_0 , R_1 ,

where R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

11. (Currently Amended) A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i , ~~the~~ method comprising the following steps:

Initialisation:

$R_0 <- 1; R_1 <- A; i <- M-1; k <- 1$

As long as $i \geq 0$, repeat the common-block ($F(k, s)$) set of instructions:

$k <- (D_i) \text{ AND } (/k)$

$\gamma'(s, k): R_0 <- R_0 \times R_k$

$i <- i - (/k)$

Return R_0 ; R_0 ,

where R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

12. (Currently Amended) A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i , ~~the~~ method comprising the following steps:

Initialisation:

$R_0 <- 1; R_1 <- A; i <- 0; k <- 1$

As long as $i \leq M-1$, repeat the block $\langle F(k, s) \rangle$ common set of instructions:

$$k \leftarrow k \oplus D_i$$

$$\gamma(k): R_k \leftarrow R_k \times R_1$$

$$i \leftarrow i+k$$

Return R_0, R_1

where R_0 and R_1 are values stored in two registers, respectively, and

k is a loop pointer indicating a number of executions performed with the

common set of instructions.

13. (Currently Amended) A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i , the method comprising the following steps:

Initialisation:

$$R_0 \leftarrow 1; R_1 \leftarrow A; R_2 \leftarrow A^3;$$

$$D_{-1} \leftarrow 0; i \leftarrow M-1; s \leftarrow 1$$

As long as $i \geq 0$, repeat the block $\langle F(k, s) \rangle$ common set of instructions:

$$k \leftarrow (s)x(k+1) + sx(D_i + 2x(D_i \text{ AND } D_{i-1}))$$

$$s \leftarrow /((k \bmod 2) \oplus (k \bmod 4))$$

$$\gamma(k,s): R_0 \leftarrow R_0 \times R_{sx(k \bmod 2)}$$

$$i \leftarrow i - sx(k \bmod 2 + 1)$$

Return R_0, R_1

where R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

14. (Currently Amended) A method according to Claim 9, used in the implementation of an exponentiation calculation of the type $B = A^D$, with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i, ~~the method~~ comprising the following steps:

Initialisation:

$$R_0 \leftarrow 1; R_1 \leftarrow A; R_2 \leftarrow A^3;$$

$$D_{-1} \leftarrow 0; i \leftarrow M-1; s \leftarrow 1$$

As long as $i \geq 0$, repeat:

$$k \leftarrow (s/x)(k+1)$$

$$s \leftarrow s \oplus D_i \oplus ((D_{i-1} \text{ AND } (k \bmod 2))$$

$$F(k,s): \quad R_0 \leftarrow R_0 \times R_{kxs}$$

$$i \leftarrow i - kxs - (D_i)$$

Return R_0, R_1 ,

where R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

15. (Currently Amended) A method according to ~~one of Claims 7 or 8, in which the links between claim 7, wherein at least two of the following data items, (a)~~ the value of the ~~a~~ loop pointer (k) and/or ~~(b)~~ the value of the ~~a~~ state pointer (s) and/or ~~(c)~~ the value of the input variable (D_i) and/or ~~(d)~~ the number of instructions of the selected block of instructions, (Π_i) associated with the value of the input data item (D_i) are linked and such linking is defined by a table with several inputs such as a matrix ($U(k,1)$).

16. (Currently Amended) A method according to Claim 15, used in the implementation of an exponentiation calculation of the type $B = A^D$, with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i , ~~the method comprising the following step:~~

As long as $i \geq 0$, repeat the block ~~($E(k,s)$) common set of instructions:~~

$$k \leftarrow (/s)x(k+1) + sx2x(/D_i)$$

$$s \leftarrow U(k,1)$$

$$\gamma(k,s): \quad R_0 \leftarrow R_0 \times R_{U(k,0)}$$

$$i \leftarrow i - s$$

where $(U(k,1))$ is the following matrix:

$$(U(k,1)) \begin{matrix} 0 \leq k \leq 2 \\ 0 \leq i \leq 1 \end{matrix} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix},$$

R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

17. (Currently Amended) A method according to Claim 15, used in the implementation of an exponentiation calculation of the type $B = A^D$ according to the algorithm (M, M^3) , with D being an integer number of M bits, and each bit (D_i) of D corresponding to an input variable of input index i , ~~the method~~ comprising the following step:

As long as $i \geq 0$, repeat the common-block $(\Gamma(k, s))$ set of instructions:

$$k \leftarrow (s)x(k+1) + sx(D_i + 2x(D_i \text{ AND } D_{i-1}))$$

$$s \leftarrow U(k, 2)$$

$$\gamma(k, s): \quad R_0 \leftarrow R_0 \times R_{U(k, 0)};$$

$$i \leftarrow i - U(k, 1)$$

where $(U(k, 1))$ is the following matrix:

$$(U(k, 1)) \begin{matrix} 0 \leq k \leq 5 \\ 0 \leq i \leq 2 \end{matrix} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 2 & 1 \end{pmatrix},$$

R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

18. (Currently Amended) A method according to Claim 15, used in the implementation of a calculation on an elliptic curve in affine coordinates, a calculation using operations of the addition or doubling of points type, and in which the following step is performed:

As long as $i \geq 0$, repeat $\Gamma(k,s)$:

$\gamma(k): R_{U(k,0)} \leftarrow R_1 + R_3;$
 $R_{U(k,1)} \leftarrow R_{U(k,1)} + R_{U(k,2)};$
 $R_5 \leftarrow R_2/R_1; R_{U(k,3)} \leftarrow R_1 + R_5;$
 $R_{U(k,4)} \leftarrow R_5^2;$
 $R_{U(k,4)} \leftarrow R_{U(k,4)} + a;$
 $R_1 \leftarrow R_1 + R_{U(k,5)};$
 $R_2 \leftarrow R_1 + R_{U(k,6)}; R_6 \leftarrow R_1 + R_{U(k,7)};$
 $R_5 \leftarrow R_5 \cdot R_6; R_2 \leftarrow R_2 + R_5$
 $s \leftarrow k - D_i + 1$
 $k \leftarrow (k+1) \times (1/s);$
 $i \leftarrow i - s;$

where $(U(k,1))$ is the following matrix:

$$(U(k,1)) = \begin{pmatrix} 1 & 2 & 4 & 1 & 6 & 6 & 4 & 3 \\ 6 & 6 & 3 & 5 & 1 & 5 & 2 & 6 \end{pmatrix} [[.]]_1$$

R_0 and R_1 are values stored in two registers, respectively,

k is a loop pointer indicating a number of executions performed with the common set of instructions, and

s is a state pointer indicating whether the predefined number has been reached.

19. (Currently Amended) A method for obtaining an elementary block $\langle F(k,s) \rangle$ set of instructions common to ~~N~~ a plurality of predefined blocks of instructions $\langle H_1, \dots, H_N \rangle$, a method able to be used for implementing a cryptographic calculation method according to ~~one of Claims 1 to 12, the method being characterised in that it comprises claim 21, comprising~~ the following steps:

E1: breaking down each predefined block of instructions $\langle H_1, \dots, H_N \rangle$ into a series of elementary blocks $\langle \gamma \rangle$ that are equivalent vis-à-vis in the context of a covert channel attack, and classifying all the elementary blocks,

E2: seeking identifying a common elementary block $\langle \gamma(k,s) \rangle$ that is equivalent to all the elementary blocks $\langle \gamma \rangle$ of all the predefined blocks of instructions,

E3: seeking identifying a common block $\langle F(k,s) \rangle$ comprising at least the common elementary block $\langle \gamma(k,s) \rangle$ previously obtained identified and an instruction to update a loop pointer $\langle k \rangle$ such that an execution of the common elementary block associated with the value of the loop pointer $\langle k \rangle$ and an execution of the elementary block with a rank equal to the value of the loop pointer $\langle k \rangle$ are identical.

20. (Currently Amended) A method according to Claim 19, characterised in that wherein, during step E1, at least one fictional instruction is added to at least one predefined block of instructions.

21. (New) A method for implementing a cryptographic calculation in an electronic device, comprising the following steps:

selecting a block of instructions from amongst a plurality of predefined blocks of instructions, as a function of an input variable; and

executing a set of instructions that is common to the plurality of predefined blocks of instructions a predefined number of times, wherein said predefined number is associated with the selected block of instructions.

22. (New) A method according to claim 5, wherein the value of the state pointer is a function of the value of the input variable and/or of the number of instructions in the selected block of instructions.

23. (New) A method according to claim 15, wherein said several inputs comprise a matrix.

24. (New) A method according to claim 21, wherein said electronic device is a chip card.